PARALELIZAÇÃO EM QUÍMICA

Nelson Henrique Morgon
Instituto de Química - Universidade Estadual de Campinas - CP 6154 - 13083-970 - Campinas - SP

Recebido em 17/11/94; aceito em 9/2/95

Parallel computation applied to theoretical chemistry has increased very much in the last 15 years. In this it work is presented results employing parallel algorithms in the calculation of electronic repulsion integrals. Although the improvement of the performance is almost obvious, it was observed that the random expansion of the number of CPUs does not necessarily provides the best performance of the calculations. General features about parallel computation and other *ab initio* calculations are presented.

Keywords: parallel algorithms; tools to parallelization; *ab initio* calculations.

INTRODUÇÃO

A utilização de métodos de cálculos baseados na química quântica computacional tem-se tornado uma ferramenta de grande importância para o estudo de problemas de estrutura molecular, estabilidade e mecanismo de reação. As técnicas fundamentais, como a teoria do orbital molecular, datam dos primeiros dias da mecânica quântica, há mais de meio século. Contudo, aplicações quantitativas em larga escala, só foram possíveis recentemente em função do explosivo desenvolvimento dos computadores (*hardware*) associado a algoritmos matemáticos eficientes[1].

O estudo de sistemas químicos pode ser tratado teoricamente, utilizando-se metodologias com origem na equação de Schroedinger: método *ab initio*, onde não são feitas aproximações para as integrais ou hamiltonianos eletrônicos[2] e semiempírico, onde simplificações na parte eletrônica são feitas usando parametrizações obtidas através de dados experimentais e, métodos empíricos, construídos utilizando-se informações experimentais (p. ex., método de mecânica molecular, onde equações matemáticas são parametrizadas empregando-se constantes de força obtidas a partir de espectros vibracionais). O compromisso na obtenção de resultados precisos implica em um maior custo computacional, de modo que cálculos *ab initio* de alta qualidade limitavam-se até pouco tempo a sistemas químicos menores. Para avaliar a extensão deste custo computacional, vale dizer que em uma das aproximações mais simples para o tratamento *ab initio* de átomos e moléculas utilizando o formalismo da teoria do orbital molecular, o método Hartree-Fock (HF), o tempo computacional necessário gasto no estudo de um sistema contendo n-elétrons descrito por N funções de base é proporcional a N^4. Tratamentos mais rigorosos que corrigem algumas das aproximações do método HF, como a introdução de efeitos de correlação eletrônica (p. ex., CI - *Configuration Interaction*, MBPT - *Many-Body Perturbation Theory*, MCSCF - *MultiConfiguration Self-Consistent Field* e outros), estão na ordem de N^5 a N^6. Assim, se para sistemas "pequenos", como átomos leves e moléculas simples, é de se imaginar a necessidade de grandes recursos computacionais para a obtenção de resultados comparáveis a dados experimentais, para o tratamento de sistemas com 50, 100 ou mais átomos, como em sistemas orgânicos, fármacos, complexos inorgânicos, ou sistemas biológicos, este esforço é muito maior.

O aumento no desempenho computacional obtido com equipamentos mais modernos e o acesso a programas mais eficientes empregados no estudo de estrutura eletrônica de sistemas atômicos e moleculares, têm permitido hoje no país, um intercâmbio cada vez maior entre grupos de pesquisas em química nas áreas experimental e teórica, tornando possível a tarefa de estudar teoricamente sistemas químicos freqüentemente encontrados nos laboratórios de pesquisa. Assim, pode-se em muitas situações, utilizando-se cálculos teóricos, estudar sistemas moleculares complexos e obter resultados que apresentam uma boa correlação com dados experimentais. Conseqüentemente, estes resultados possibilitam uma compreensão mais profunda da natureza de importantes eventos químicos através da interpretação de aspectos estrutrurais em mecanismos de reação, cálculos de propriedades eletrônicas e termodinâmicas, etc. Auxiliam nestas tarefas, ferramentas de visualização de estruturas moleculares, de densidades eletrônicas e técnicas de paralelização, que ao proporcionar a diminuição do tempo computacional, abrem perspectivas para extender os cálculos, com maior refinamento, para sistemas químicos ainda maiores.

O objetivo principal deste trabalho é fazer uma apresentação de um dos grandes avanços computacionais, ou seja, o uso de programação paralela em química, em função principalmente das potencialidades indiscutíveis que ela possui e da crescente difusão que ela vem tendo nos últimos quinze anos. Deseja-se também abordar algumas técnicas, termos e outros aspectos da programação paralela.

Para dar noção do desenvolvimento da computação paralela, basta salientar que em 1955 a IBM lançou o computador IBM/704. Este é um marco por se tratar da primeira máquina comercial que possui *hardware* capaz de executar aproximadamente 5.000 instruções por segundo (5 *kFLOPS*). Ao longo do tempo houve uma evolução significativa no desempenho dos computadores seriais, vetoriais e paralelos, ao ponto de em 1993 a *Cray Research* apresentar o sistema de processamento massivamente paralelo (*MPP - Massively Parallel Processing*) denominado CRAY T3D, onde o modelo mais avançado sendo composto de 2.048 processadores tem capacidade de processamento de pico de 307,2 *GFLOPS*. Recentemente o *CORNELL THEORY CENTER* (EUA) adquiriu um sistema massivamente paralelo (SP2) composto de 512 processadores e capaz de efetuar 136 bilhões de cálculos por segundo. Por estes números nota-se que em cerca de 40 anos houve uma evolução em mais de 60 milhões de vezes no desempenho computacional. Na previsão de Harrison e Bair[3] existem vários projetos para 1995 onde computadores massivamente paralelos constituídos por milhares de processadores possam atingir velocidades superiores a 1 *TFLOPS*.

Arquitetura e programação paralela

As aplicações científicas atualmente têm exigido um aumento na velocidade computacional, assim grandes esforços têm sido feitos no sentido de suprir esta tendência. E, sem dúvida alguma, o desenvolvimento de arquiteturas baseadas no conceito de processamento paralelo tem contribuído, e muito para isto. De um modo geral, um sistema paralelo consiste de vários processadores e unidades de memória, além de outros recursos compartilhados trabalhando-se ao mesmo tempo em um mesmo problema. Uma analogia com processamento paralelo pode ser feita imaginando-se que uma tarefa dividida em *n*-subtarefas, e executada simultaneamente por *n*-pessoas, gastaria a *n*-ésima parte do tempo necessário se a mesma fosse executada por apenas uma pessoa. Isto mostra que a execução simultânea de subtarefas é um conceito de paralelização de trabalho bastante familiar. O efeito líquido deste processamento paralelo é uma substancial redução no tempo de solução do problema[4-7]. O processamento paralelo apresenta desempenho superior em relação às técnicas convencionais (processadores seriais ou seqüenciais).

Diante da grande variedade de arquiteturas em paralelo, elas podem ser classificadas em duas principais categorias: SIMD - (*Single Instruction/Multiple Data*), onde todos os processadores trabalham efetuando instruções idênticas em diferentes blocos de dados e MIMD - (*Multiple Instruction/Multiple Data*). A arquitetura MIMD é composta de computadores onde as CPUs são independentes e a comunicação é feita por fios (*message passing*) - os computadores com memória distribuída ou onde a comunicação é feita a partir de uma região comum da memória (*shared memory*) - os computadores com memória compartilhada.

As arquiteturas de memória compartilhada possuem um número determinado de processadores operando sob o controle de um fluxo de instruções simples distribuído por uma unidade central de controle. Cada processador tem uma pequena memória privada para armazenar programas e dados. Executam operações sincronizadas, ou seja, durante uma dada unidade de tempo, um número selecionado de processadores são ativados e estes fazem a mesma instrução, cada qual com um conjunto diferente de dados, mantendo-se os processadores remanescentes inativos. A troca de dados se dá ou por rede de interconexão ou por uma memória comum compartilhada[8]. Nas arquiteturas de memória distribuída, os processadores possuem contadores de instrução independentes, isto é, em um sistema de *p* processadores, cada um deles é um computador completo, com seu próprio controle, memória e unidades aritmética e lógica, agindo independentemente sobre um conjunto de dados[7-10].

Comparativamente as máquinas de memória compartilhada apresentam como vantagens: a) o baixo custo, b) a grande quantidade de *software* disponível e c) facilidade para programação, porém apresentam "escalabilidade"[11] limitada e o paralelismo necessita de muitas instruções (*coarse-grained parallelism*). Já os multicomputadores possuem "escalabilidade" linear e necessitam para tarefas paralelas, um menor número de instruções (*fine-grained parallelism*). Porém, como inconvenientes são difíceis de programar e em função disto possuem pouco suporte de *software*, além de serem relativamente caros[12].

O arranjo envolvido na interconexão dos processadores (ou topologia da rede) determina o desempenho do sistema como um todo e pode ser de dois tipos[13]:

- topologias dinâmicas, onde as conexões são estabelecidas via canais comuns ou via rede (redes de estações de trabalho, *clusters*, ...) com interfaces de comunicação via Ethernet, Token-ring, FDDI (*Fiber Distributed Data Interface*), SOOC (*Serial Optical Channel Converter*), ... e
- topologias estáticas, onde existem *links* entre os elementos de processamento, como as arquiteturas em árvores, anéis, hipercubos, etc. A comunicação é feita através de *links* internos, como por exemplo *High Performance Switch*, e outros.

Junto com o surgimento destas novas arquiteturas computacionais e de *hardware*, houve também a necessidade do desenvolvimento de programação, visando um melhor aproveitamento destes recursos[8,9], como:

a. **programação em paralelo**, ocorre quando da utilização de sistemas constituídos de vários processadores alocados em uma região, com uma pequena distância entre eles. A comunicação entre os processadores é extremamente rápida,
b. **programação distribuída**, utilizada em sistemas onde os processadores estão distantes uns dos outros. Geralmente a comunicação interprocessador é mais problemática, havendo a necessidade de transmissão de dados e informações via uma rede de comunicação e
c. **programação concorrente**, desenvolvida para o uso em computadores de arquitetura tradicional. Concentra-se em características de linguagem de alto nível. Linguagens de programação seqüenciais típicas (FORTRAN, PASCAL, ...) consistem de seqüências de declarações que são geralmente executadas na ordem em que são escritas. As características especiais desta nova linguagem são usadas ou para introduzir execução condicional ou repetitiva de algumas declarações ou para agrupar seqüências de declarações em procedimentos ou subrotinas. Entre estas linguagens de alto nível pode-se citar: HPF (*High Performance Fortran*), Pfortran (*Parallel Fortran*) e Maisie - linguagem de programação paralela baseada em C[14].

Para facilitar o desenvolvimento de programas paralelos encontram-se disponíveis um grande número de pacotes de sistemas comerciais e principalmente de domínio público[15] baseados em *message passing*. Dentre os mais populares podem ser citados os pacotes de *software*: LINDA[16], PVM[17], TCGMSG[18], EXPRESS[19], PARMACS[20], PICL[21], P4[22], etc. Além de ferramentas de monitoramento e visualização, linguagens de programação, tais como: HeNCE (*Heterogeneous Network Computing Environment*) - ambiente gráfico de programação paralela que possibilita um modo fácil de criar, interfaciar, compilar, executar e "debugar" programas paralelos, ParaGraph e Xab - sistemas de "*display*" gráfico para visualização do comportamento e desempenho de programas paralelos, ADAPTOR (*Automatic Data Parallelism TranslaTOR*) que transforma programas paralelos de dados escrito em Fortran com extensões de vetores, matrizes, *loops* paralelos em programas paralelos com *message passing* explícita, entre muitos outros. Além de compiladores específicos como LCAP e PFP encontrados em sistemas IBM modelo 3090.

Paralelização em química

O espectro de aplicação destes recursos computacionais nas várias áreas do conhecimento humano e particularmente em química é bastante amplo. Em química teórica pode-se citar: cálculos de estrutura eletrônica *ab initio* para grandes sistemas moleculares e/ou cálculos com alta qualidade, simulação de fases condensadas[23], Monte Carlo quântico, modelagem molecular[24], cálculos de defeitos de redes em materiais cristalinos[25], etc.

Para se ter idéia, basta salientar que simulação utilizando dinâmica molecular (DM) é um importante meio para estudar fenômenos macroscópicos em nível microscópico, porém exige uma grande demanda computacional. Isto se deve ao fato de que uma grande parte (geralmente mais de 90%) do tempo computacional para DM é gasto no cálculo de contribuições de energia e forças entre os pares atômicos não-ligados e ligados ao H. Paralelização pode deste modo ser facilmente efetuada distribuindo a lista de pares de átomos sobre um número de processadores e somando-se ao final de cada ciclo as contribuições para energia e forças[26]. Assim, a implementação de algoritmos em redes de processadores paralelos[27] com desenvolvimento

de esquemas computacionais para grandes simulações de dinâmica molecular em líquidos[28,29] são cada vez mais freqüentes.

O processamento paralelo verifica-se também no tratamento de grandes sistemas moleculares, com o desenvolvimento de algoritmos usando o Método Monte Carlo para análise conformacional de polipeptídeos[30], algoritmos paralelo para cálculos de campo de força em macromoléculas[31], determinação estrutural de proteínas[32], etc.

Embora, cada vez mais seu uso seja corrente, paralelismo não é tão recente na química computacional, o embrião de sua utilização em Química Quântica surgiu em meados da década de 80 nos estudos energéticos de pontes de hidrogênio em pares de bases de DNA, onde Clementi *et alli*[33,34], utilizavam sistemas com 6 a 10 processadores paralelos. Vale salientar que se vetorização for considerada como um caso especial de paralelismo, então seu início é um pouco anterior.

Se por um lado o surgimento desta nova geração de computadores permitiu um grande avanço no estudo teórico de estruturas eletrônicas de sistemas moleculares, tanto em relação ao tamanho dos sistemas moleculares e aumento do número de funções no conjunto de base, quanto ao refinamento e exatidão buscadas nos resultados, por outro lado programas computacionais empregados nestes estudos e que utilizam-se de computadores de arquitetura paralela, (redes e *clusters* de estações de trabalho, multiprocessadores, sistemas massivamente paralelos, ...) tornam-se mais e mais freqüentes, como exemplo: DISCO[35], GAMESS[36], HONDO[37], TURBOMOLE[38], AMBER[39], entre outros.

Metodologias que têm adquirido grande sucesso atualmente em função das vantagens que possuem, são as que baseiam-se na Teoria do Funcional de Densidade (TFD)[40] (ou *DFT - Density Functional Theory*). Potencialmente a TFD é bastante adequada para o uso de programação paralela. Deste modo existem grandes perspectivas que a conjunção destas técnicas possam ser aplicadas no estudo de grandes sistemas moleculares.

A implementação de algoritmos paralelos em programas de cálculos de estrutura eletrônica pode ser feita para o cálculo de integrais de um e dois elétrons[5,41], em cálculo de componente de energia tripla de quarta ordem em MBPT[42], no cálculo SCF-direto[35,43-45], em método de pseudopotencial para cálculo de energia total[46], no método de correlação eletrônica de *Coupled-Cluster* com excitações simples e duplas (CCSD)[6], no método de diagonalização de Jacobi[4], em cálculos de gradientes nucleares e energia SCF direto[47] e em várias outras aplicações.

METODOLOGIA COMPUTACIONAL

Cálculo paralelizado de integrais de repulsão eletrônica

Em cálculos de orbitais moleculares em nível Hartree-Fock, a obtenção das integrais de repulsão eletrônica é, sem dúvida, a etapa de cálculo que consome mais tempo de processamento (tempo de CPU), principalmente as integrais de multicentros. Assim, o cálculo simultâneo de integrais, feitos em diferentes CPUs, tende a minimizar o tempo total de cálculo.

Neste trabalho, fez-se um estudo de alguns sistemas moleculares utilizando-se paralelização no cálculo das integrais de repulsão eletrônica em um código *ab initio* através de programação distribuída MIMD, com modelo de distribuição MPMD (*Multi Program/Multiple Data*). Modelo de distribuição em paralelo composto por vários subprogramas, onde cada um deles destina-se a uma determinada tarefa, ficando distribuídos pelos processadores previamente definidos. Na distribuição SPMD (*Single Program/Multiple Data*), existe apenas um programa global, alocando-se em cada processador uma cópia do mesmo. Programas paralelos que utilizam rotinas do pacote TCGMSG[18], como GAMESS[36], encontram-se nesta categoria.

Para o cálculo de integrais de repulsão eletrônica em um código *ab initio* Hartree-Fock, utilizou-se a computação paralela de memória distribuída através de PVM (*Parallel Virtual Machine*)[17] versão 2.4.1. PVM é um pacote de *software* de domínio público, desenvolvido no Laboratório Nacional de Oak Ridge (EUA) e suportado em sistemas UNIX. A conceituação de PVM está baseada na simplificação da programação em paralelo e na facilidade de envio e recebimento de dados entre diferentes CPUs. É um sistema *message passing*, que permite rodar códigos em paralelo num ambiente heterogêneo de máquinas conectadas por uma ou mais redes de comunicação, incluindo sistemas de CPU simples, máquinas vetoriais e multiprocessadores (desde computadores pessoais simples PCs, até supercomputadores de última geração (p. ex: CRAY T3D, SP2 e outros). Aplicação recente utilizando o pacote PVM como ferramenta de paralelização pode ser encontrada no trabalho de Ludëk e Koca[48] sobre a elucidação de hipersuperfície de energia potencial conformacional em modelagem molecular, utilizando uma rede de estações de trabalho. Embora em PVM a memória necessária para armazenar programas individuais seja sensivelmente menor se comparada à necessária para um programa global, a comunicação, a sincronização e o controle das tarefas executadas pelos processadores (ou processos) têm que ser feitas por um processo *daemon* (denominado **pvmd**)[49] residente em todos os computadores da rede.

O fato de se ter um programa menor armazenado na memória do processador torna-se bastante vantajoso, uma vez que as tarefas executadas por este programa no processador podem ser feitas diretamente na memória disponível da máquina. Existe um aumento de ganho sensível na execução dos cálculos, principalmente quando se está trabalhando em máquinas com tamanho limitado de memória. Pode-se ainda deixar a tarefa de leitura/escrita de dados/resultados a cargo do programa principal (denominado *host*) encarregado da divisão e distribuição das tarefas aos programas auxiliares (denominados *nodes*). Outro aspecto que merece ser discutido dentro do modelo MIMD está relacionado à não-sincronização dos processadores. A vantagem da não-sincronização é devida a uma maior flexiblidade na resolução de um conjunto de subtarefas pelos processadores, pois se um determinado processo terminou sua subtarefa, independentemente do término de outro processo, ele pode começar outra subtarefa e assim agilizar o término da tarefa global. O inconveniente disto, se dá em função da maior dificuldade na programação. Dentro do PVM existem rotinas apropriadas que permitem controlar e sincronizar o fluxo de transmissão de mensagens entre os processadores.

As figuras 1a e 1b mostram modelos de programas (*host* e *node*) escritos em Fortran utilizando-se funções do PVM versão 2.4.1. Vê-se pelas figuras 1 a independência dos programas, pode-se alocá-los individualmente em processadores separados, sem perda substancial de desempenho, pois o processo de transmissão de mensagens é bastante eficiente.

Neste trabalho empregou-se uma rede local de computadores (PCs, estações de trabalho de arquitetura SUN4 e IBM RISC/6000 modelos 320H, 370 e 550) do Instituto de Química conectada a um *cluster* de máquinas POWER RISC/6000 560 e ao complexo SP1 do Centro Nacional de Processamento de Alto Desempenho (CENAPAD-SP) da UNICAMP, como pode ser visualizado pela figura 2.

A parte paralelizada do programa foi apenas a do cálculo de integrais de repulsão eletrônica, os passos neste processo de paralelização são dados por:

1. *loop* sobre os índices *i, j, k* e *l* satisfazendo-se a ordenação: $i \geq j$, $k \geq l$ e $[ij] \geq [kl]$, para $[ij] = \frac{1}{2} i(i - 1) + j$[50];
2. distribuição dos pacotes contendo os 4 índices que definem as **n** integrais - onde **n** é o número total de integrais pelo número de processos utilizados - para o *i*-ésimo processo. Assim num sistema onde existam 1.000.000 de integrais, distribuídas em 5 processos, tem-se o cálculo simultâneo de 200.000 integrais em cada processo, e

(a)

```
      PROGRAM HOSTPROGRAM
      INTEGER I, INFO, NPROC, MSGTYPE, MYNUM, INST(4), ...
      DOUBLE PRECISION RESULT(4), DATA(100), ...
C     ENROLL THIS PROGRAM IN PVM (VERSION: 2.4.1)
      CALL FENROLL( "HOSTPROGRAM\0", MYNUM)
C     INITIATE NPROC INSTANCES OF NODE PROGRAM
      NPROC = 4
      ARCH  = "\0"
      DO 10 I=1,NPROC
      CALL FINITIATE( "NODEPROGRAM\0", ARCH, INST(I) )
10    CONTINUE
C ------- BEGIN USER PROGRAM --------
            :
C     BROADCAST DATA TO ALL NODE PROGRAMS
      CALL FINITSEND()
      CALL FPUTNDFLOAT( DATA, 100, INFO )
      MSGTYPE  = 1
      CALL FSND( "NODEPROGRAM\0", -1, MSGTYPE, INFO )
            :
C     WAIT FOR RESULTS FROM NODES
      MSGTYPE  = 2
      DO 20 I=1,NPROC
      CALL FRCV( MSGTYPE, INFO )
      CALL FGETNDFLOAT( RESULT(I), 1, INFO )
20    CONTINUE
            :
C --------- END USER PROGRAM --------
C     PROGRAM FINISHED LEAVE PVM BEFORE EXITING
      CALL FLEAVE()
      STOP
      END
```

(b)

```
      PROGRAM NODEPROGRAM
      INTEGER  INFO, MYNUM, HOSTNUM, BYTES, MSGTYPE, ...
      DOUBLE PRECISION RESULT, DATA(100), ...
      CHARACTER*16 HOST, ...
C     ENROLL THIS PROGRAM IN PVM  (VERSION: 2.4.1)
      CALL FENROLL( "NODEPROGRAM\0", MYNUM )
C ------- BEGIN USER PROGRAM --------
            :
C     RECEIVE DATA FROM HOST
      MSGTYPE  = 1
      CALL FRCV( MSGTYPE, INFO )
      CALL FGETNDFLOAT( DATA, 100, INFO )
      CALL FRCVINFO( BYTES, MSGTYPE, HOST, HOSTNUM, INFO )
            :
      RESULT = USER_ROUTINE( DATA )
C     SEND RESULT TO HOST
      CALL FINITSEND()
      CALL FPUTNDFLOAT( RESULT, 1, INFO )
      MSGTYPE  = 2
      CALL FSND( "HOSTPROGRAM\0", HOSTNUM, MSGTYPE, INFO )
            :
C --------- END USER PROGRAM --------
      CALL FLEAVE()
      STOP
      END
```

Figura 1. Modelo de programas (a) host *e (b)* node *em Fortran usando-se rotinas PVM.*

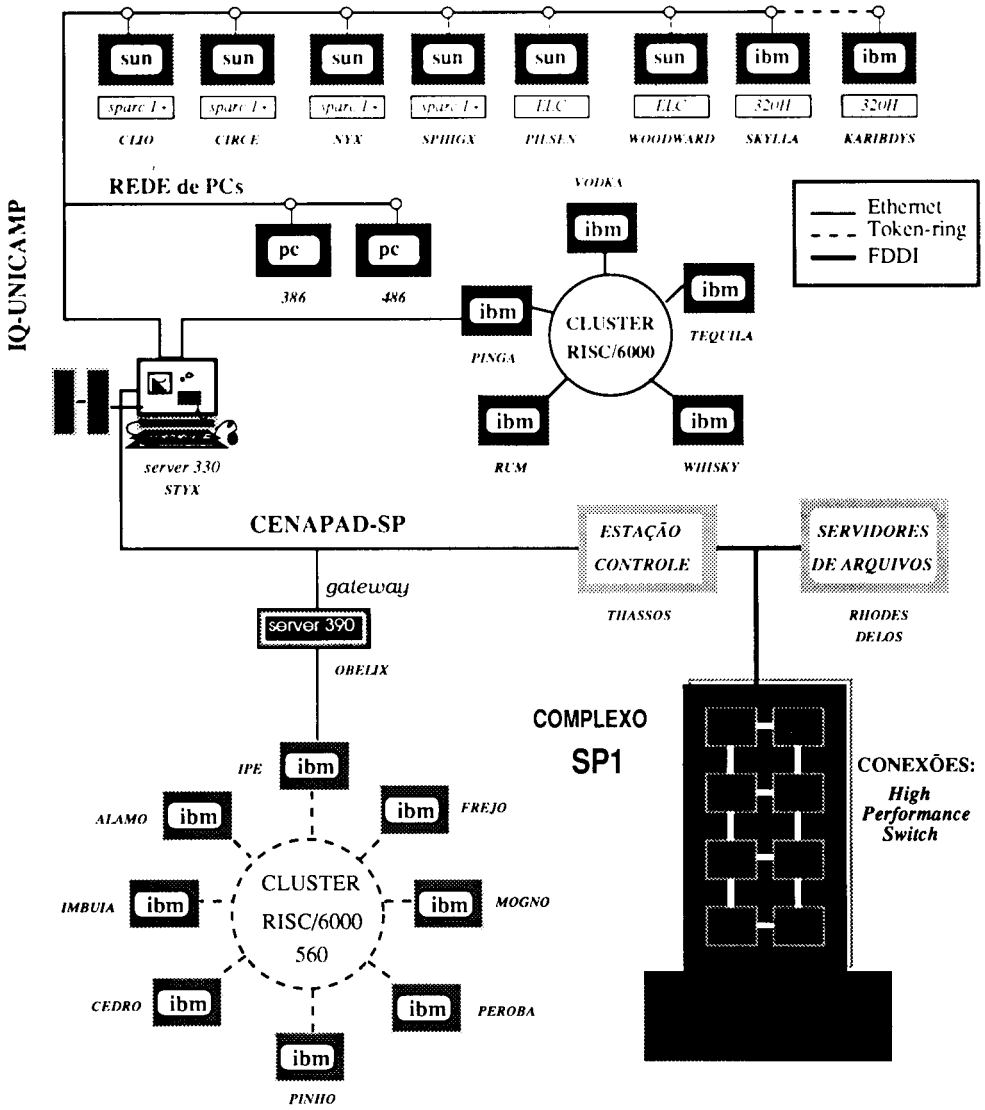

Figura 2. *Diagrama da rede de computadores do Instituto de Química conectada ao Centro Nacional de Processamento de Alto Desempenho (CENAPAD-SP) localizado na Universidade Estadual de Campinas (UNICAMP) e utilizada neste trabalho.*

3. recebimento e armazenamento das integrais calculadas, a serem utilizadas posteriormente.

A figura 3 apresenta um esquema deste procedimento. Em **0** tem-se o programa principal que chama a rotina **1**, responsável pela obtenção dos pacotes contendo os quatro índices que definem as integrais de repulsão eletrônica, **< ij | kl >**, e pela distribuição destes pacotes entre as diversas CPUs (representadas por **2**, **3**, **4**, **5**, **6** e ...) que irão calcular as respectivas integrais e retornarão os resultados para serem armazenados em disco pela rotina **1**. Observa-se que apenas um processo, definido por **1**, controla a escrita dos resultados das integrais em disco. Isto permite a independência entre os diversos processos envolvidos e mantém um fluxo constante de envio e recebimento de informações.

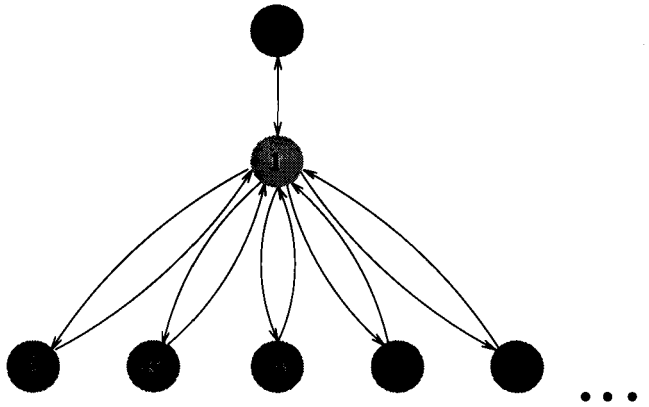

Figura 3. Esquema de paralelização utilizado no cálculo das integrais de repulsão eletrônica.

Velocidade de transmissão de dados: nfs X paralelismo

O uso de estações de trabalho tem-se tornado bastante popular nos laboratórios computacionais. Fatores como o baixo custo e manutenção em relação a computadores maiores têm contribuído para isto. Tem-se verificado sua utilização em várias atividades científicas; na Química, por exemplo, uma de suas principais aplicações se dá em cálculos de estruturas eletrônicas. A grande e talvez principal vantagem destes equipamentos está no uso de sistemas de arquivos em rede (*NFS - Network File System*), permitindo que estações de trabalho, denominadas clientes, tenham acesso transparente sobre uma rede. Assim, uma estação cliente pode operar em arquivos que residem em uma variedade de máquinas servidoras e através de uma variedade de sistemas operacionais. Com as redes é possível um ganho substancial de espaço em disco e memória. Este ganho pode ainda ter um desempenho maior se entre outras técnicas, utilizar-se paralelização.

A verificação da eficiência de transmissão de dados via NFS contra paralelismo foi feita através de um programa empregando-se o pacote PVM. Analisou-se a velocidade de transmissão de diferentes conjuntos de dados (de 100 a 1.000.000 *bytes*) de estações clientes para estações servidoras na rede de arquitetura paralela (Fig. 3).

RESULTADOS

A análise do tempo de CPU gasto no cálculo das integrais de repulsão eletrônica no código utilizado mostra que elas consomem a maior parte do tempo total de um cálculo para a energia Hartree-Fock. Na tabela 1 tem-se os tempos gastos nas várias etapas do cálculo da energia Hartree-Fock para a molécula HF na geometria molecular de equilíbrio ($R_{H\text{-}F}$ = 1,74 Å) com conjunto de funções de base 6-31G. Cerca de 95% do tempo total para o cálculo desta energia foi gasto no cômputo das integrais de repulsão eletrônica.

Tabela 1. Tempos de CPU (em segundos) gastos nas diversas etapas do cálculo da energia total SCF da molécula HF (R = 1,74 Å) usando-se funções de base 6-31G.

Etapas	Tempo[a]
Integrais de 1-elétron	2,41
Integrais de 2-elétrons	119,80
Auto consistência (scf)	2,42
Outras	0,38
Total	**125,01**

[a]Resultados obtidos utilizando um SPARCserver modelo 330 (24 Mb de memória RAM).

A tabela 2 apresenta os resultados em nível Hartree-Fock utilizando funções de base 6-31G para uma série de moléculas e mostra a redução obtida entre os tempos gastos para os diferentes tipos de cálculos, sem e com paralelização. A redução no tempo gasto usando-se paralelização para estes sistemas ficou em torno de 30 a 40%. Estes números, é claro, variarão de acordo com as disponibilidades e configurações dos computadores utilizados, como pode ser visto por esta tabela.

Pelas figuras 4a e 4b verifica-se o desempenho do cálculo das integrais de repulsão eletrônica para as moléculas CH_4 e CH_2=O, na geometria otimizada usando conjuntos de base 6-31G e 6-31G**, respectivamente. Nota-se que para diferentes configurações da rede, este desempenho não está diretamente relacionado com o aumento do número de CPUs. A adição de

Tabela 2. Tempos de CPU (em segundos) gastos no cálculo de integrais de repulsão eletrônica de algumas moléculas, usando-se funções de base 6-31G, nas geometrias moleculares de equilíbrio. Cálculos seqüenciais e paralelos via PVM.

Moléculas	Tipo de cálculo[a]		NOA[b]	NFB[c]	Nº de Integrais:	
	Seqüencial	Paralelizado			Total	Int. Calc.[d]
HF	119,80	34,60	11	26	2.215	677
H_2O	206,60	64,32	13	30	4.186	2.260
NH_3	346,85	130,17	15	34	7.260	5.972
CH_4	536,38	180,91	17	38	11.781	10.695
C_5H_5N	142.837,88	47.756,22	64	152	2.164.240	1.207.354

[a]Tempo de cálculo seqüencial usando um SPARCserver 330. Cálculo paralelo em rede com 3 máquinas (1 SPARCserver 330 e 2 SPARCstations 1+ com 8 Mb de memória RAM cada uma) e o tempo de cálculo refere-se ao processo mais lento obtido entre as 3 máquinas (uma SPARCstations 1+).
[b]NOA = número de funçõs de base após contração.
[c]NFB = número de funções primitivas.
[d]Número de integrais efetivamente calculadas, cujos valores são maiores que $1{,}0\times10^{-12}$ (fator de corte).

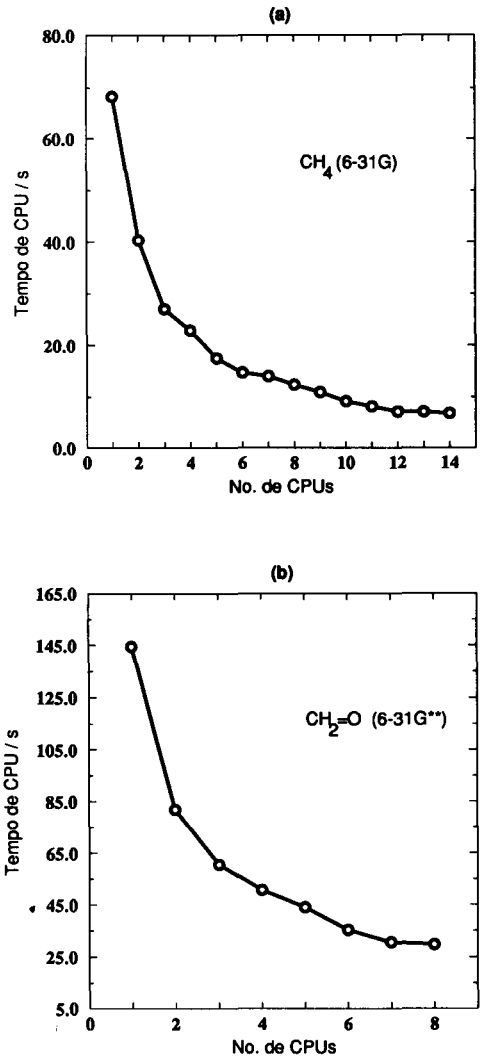

Figura 4. *Tempo (em s) gasto no cálculo das integrais de repulsão eletrônica* versus *o Número de CPUs, para (a) CH_4 e (b) CH_2=O.*

processadores por si só não altera significativamente a redução do tempo gasto. Existe uma mudança drástica de quase 50% no tempo de cálculo quando se passa de 1 para 2 CPUs, mas isto não continua acontecendo com o incremento de mais processadores. A partir da 7ª CPU adicionada, a redução do tempo de cálculo deixa de ser significativa.

No cálculo de integrais de repulsão eletrônica para o átomo de Ar usando-se conjuntos de funções de base STO (10s7p) pode-se verificar uma redução substancial do tempo total gasto, como mostra a tabela 3. Esta análise mostra um aspecto interessante: comparando-se o resultado de uma única máquina (**ipe**) com o da **rede-3** (com 15 máquinas) observa-se que os tempos de CPUs são praticamente os mesmos (4,1 e 3,9 segundos, respectivamente). Neste caso uma redução tão pequena no tempo de cálculo não justifica a utilização da rede paralela. Isto deve-se ao fato de que o tempo de comunicação entre os processadores passa a ser predominante em relação ao tempo de cálculo efetivamente gasto. O melhor desempenho é dado pela configuração da **rede-2**. Os componentes desta rede são equipamentos mais rápidos interfaciados mais eficientemente (comunicação por *token-ring*), o que torna a transmissão de mensagens entre CPUs mais rápida. Assim, deve-se ressaltar que, quando se tem uma das máquinas muito lenta, o tempo calculado fica dependente desta máquina. No caso do uso de 15 CPUs, caso uma das unidades apresente-se lenta em relação as

Tabela 3. Tempo de CPU (em segundos) gasto no cálculo das integrais de repulsão eletrônica[a] para o átomo de Ar usando-se conjunto de base de Slater (10s7p) e diferentes configurações de rede.

Configuração[b]	styx	skylla	ipe	rede-1(5)[c]	rede-2(7)	rede-3(15)
Tempo	33,63	16,12	4,11	10,56	0,83	3,88

[a]71.631 integrais calculadas.
[b]**styx**: SPARCserver 330 (24 Mb), **skylla**: RISC/6000 320H (32 Mb), **ipe**: RSC/6000 560 (32 Mb), **rede-1**: 1 SPARCserver, 2 SPARCstation 1+ e 2 RISC/6000 320H, **rede-2**: 7 RISC/6000 560 em *cluster* e **rede-3**: **rede-1** + **rede-2** + 3 SPARCstation 1+.
[c]Em parênteses está representado o número de máquinas.

outras, o tempo de cálculo torna-se maior do que quando se tem um menor número de computadores, porém mais rápidos. Portanto, o tempo gasto será dado em função desta CPU. Deve-se então, procurar fazer um balanço entre vários aspectos, afim de que se possa obter o melhor da arquitetura paralela empregada. Matematicamente, pode-se resumir o desempenho de um programa rodando em rede paralela em função da Lei de Amdahl. Em um sistema de N processadores a velocidade de processamento ideal é dada por $V_I = N$. Porém, realisticamente tem-se que levar em conta o percentual do código paralelizado, assim:

$$V_A = \frac{1}{1 - f + \frac{f}{N}} \qquad (1)$$

onde f é a fração do tempo de execução do código que é paralelizável. Por exemplo, se $f = 0,9$ (90% paralelizável) para 4 processadores, tem-se $V_I = 4,00$ e $V_A = 3,08$.

No estudo da velocidade de transmissão de dados via NFS e PVM, observa-se pela tabela 4, que quando o número de *bytes* transmitidos é pequeno não existe diferença significativa entre os dois modos de transmissão, mas quando o conjunto total de dados pode ser fragmentado em conjuntos menores, daí tem-se uma visualização clara da eficiência do paralelo. Assim, para a transmissão de 1Mb via PVM o tempo gasto é muito menor que por NFS. Estas observações são válidas tanto para uma arquitetura comum de estações de trabalho quanto para um *cluster*. O uso de uma máquina remota mostra que a troca de informações entre esta máquina cliente e a máquina servidora é relevante, quando se faz a transmissão de pequenos pacotes de dados. Pela análise da velocidade de transmissão de dados via NFS e PVM nota-se claramente que a fragmentação de grandes conjuntos de dados é mais eficientemente transmitida através de uma rede.

Tabela 4. Velocidade de transmissão (em kb/segundos) de diferentes conjuntos de dados usando-se diferentes tipos de rede e configurações.

Rede	1kb		1Mb	
	NFS	*PVM*	*NFS*	*PVM*
rede-1a[a]	0,04	0,07	47,79	49,01
rede-1b[a]	0,02	0,03	24,96	29,81
rede-2a[b]	0,04	0,02	47,79	10,04
rede-2b[b]	0,02	0,01	24,96	5,65
rede-3[c]	0,02	0,01	24,96	4,09

[a]**rede-1a** composta de 2 máquinas (1 SPARCserver 330 e 1 SPARCstation 1+) e **rede-1b** composta de 2 máquinas (2 RISC/6000 560).
[b]**rede-2a** composta de 5 máquinas (1 SPARCserver 330 e 4 SPARCstation 1+) e **rede-2b** composta de 5 RISC/6000 560.
[c]**rede-3** composta de 7 RISC/6000 560.

Análise do desempenho do SP1

Computadores paralelos têm potencial para efetuar cálculos numéricos numa relação custo/benefício muito mais efetiva que supercomputadores convencionais[26]. Entre estes computadores, um que possui grande potencial de aplicação paralela é o sistema paralelo "escalável" (*SP - Scalable POWERparallel*) que permite a adição de processadores. O processamento paralelo tanto através da utilização dos computadores em *cluster* quanto pelo complexo SP1 permite a redução substancial do tempo de cálculo e a racionalização do uso de espaço em disco. Fez-se um estudo de alguns aspectos do emprego simultâneo dos 8 processadores de um computador SP1, a fim de avaliar seu desempenho na execução de alguns cálculos comuns de estrutura eletrônica empregando-se para isto o programa GAMESS (versão 94) que permite execução paralela para: energia, gradiente numérico, hessianas numérica e analítica e cálculo CI nos métodos RHF, ROHF, GVB, UHF e MCSCF (os últimos dois à exceção para hessiana analítica), além de cálculo MP2 no método RHF.

Vale ressaltar que o programa foi utilizado na forma como é distribuído. Na análise do desempenho das diferentes arquiteturas não levou-se em conta nenhuma otimização do *software*, embora este seja um aspecto de grande importância.

Na tabela 5 são dadas as descrições dos cálculos feitos, que requerem desempenhos diferentes em função de características peculiares exploradas pelo programa. Todos os sistemas apresentados nesta tabela encontram-se na geometria molecular de equilíbrio, obtida a nível SCF com a base dada na 3ª coluna. Esta tabela permite a comparação entre os tempos de CPU gasto para 1 ou 8 processadores na execução do cálculo. De um modo geral o cálculo paralelo apresenta sempre um menor tempo com desempenhos distintos para os diferentes cálculos. Por exemplo, para o **Teste (1)** o tempo reduz-se em cerca de 74%, já para o **Teste (8)** esta redução é de aproximadamente 26%. Porém, nos métodos onde o cálculo de integrais não é a etapa determinante, por exemplo em cálculos do tipo CI e MCSCF, a fragmentação dos dados é tanta que o grande número de mensagens trocadas entre os processadores faz com que haja um aumento no tempo de cálculo quando do uso dos 8 processadores comparado ao cálculo com 1 processador apenas, por exemplo, para os **Testes (7)** e **(9)** observa-se um aumento em cerca de 70% e 80% do cálculo seqüencial, respectivamente.

Vários fatores afetam a velocidade de processamento com o aumento do número de CPUs. Os mais importantes são: (1) velocidade máxima de transferência de informação (*bandwidth*), que é um efeito proporcional ao tamanho da mensagem transferida e é dependente de *hardware* e/ou *software*, (2) tempo de enviar uma mensagem de comprimento-zero de um processador a outro (*latency*), fator este dependente do sistema operacional, (3) conexão física entre as máquinas, (4) topologia desta conexão, e outros. Deste modo, quanto mais intenso for o tráfico de mensagens (maior fragmentação dos dados) na comunicação entre os processadores, maior será a influência dos fatores acima no desempenho dos cálculos.

Na figura 5 observa-se o desempenho do número de CPUs empregadas *versus* o tempo de cálculo gasto na execução da tarefa para diferentes sistemas moleculares. Alguns aspectos são relevantes na observação desta figura. Primeiro, quando existe uma grande fragmentação dos dados, o tempo de comunicação entre os processadores (envio e recebimento de mensagens) passa a ser preponderante. Isto pode ser observado pela crescente diminuição na inclinação da curva. Por exemplo, para os dados referentes ao **Teste (3)** o aumento de 1 CPU causa uma redução de cerca de 20% no tempo de cálculo. Passando-se de 7 para o 8 CPUs, a redução no tempo de cálculo torna-se praticamente imperceptível. Assim existe uma relação custo/benefício que deve ser levada em consideração quando do aumento aleatório do número de CPUs para a execução de determinada tarefa.

Num cálculo de orbitais moleculares são geradas uma grande quantidade de integrais, que comumente são armazenadas em disco. Em função da disponibilidade de memória e disco, pode-se ou não armazená-las. Por exemplo, caso não disponha-se de disco suficiente, recorre-se ao cálculo denominado SCF-direto (para mais detalhes ver ref. [43]), onde obtém-se estas integrais a medida que são necessárias. Este é um recurso bastante útil e aliado a paralelização permite a execução de cálculos a princípio dependentes da disponibilidade de disco.

Para mostrar o desempenho de cálculos onde as integrais são calculadas e armazenadas em disco (**DISCO**), foram feitos alguns testes contrapondo os tempos de cálculos e quantidade de disco utilizada, aos resultados do cálculo direto (**DIRETO**). E também cálculos seqüenciais e em paralelo (**PAR**) para as duas possibilidades acima. A tabela 6 mostra os resultados destes testes e observa-se por estes resultados que em função da disponibilidade de memória e/ou de disco e trabalhando em seqüencial ou paralelo pode-se efetuar tarefas a um custo computacional determinado pela análise destas variáveis. O cálculo direto para o **Teste (1)** é cerca de aproximadamente 4 vezes mais lento, porém utiliza apenas 61,93 kbytes de disco, bem menos se comparado aos 5,6 Mbytes empregados no armazenamento das integrais calculadas. Análise semelhante pode ser feita para o **Teste (6)**. O uso de cálculo paralelo para o **Teste**

Tabela 5. Sistemas utilizados para analisar o desempenho de cálculos *ab initio* junto ao computador SP1 do CENAPAD-SP utilizando o programa GAMESS.

Teste	Método	Base	Número de		Tempo CPU (s)	
			OAs[a]	*FCS*[b]	*1 proc.*	*8 proc.*
(1) SiC_2H_6	RHF/SCF	*6-31G**	61	1	62,72	16,45
(2) $P_2H_4^+$	ROHF+grad.	*DH(d)*	56	1	18,44	10,24
(3) SiC_3H_8	RHF	*6-31G**	80	1	51,28	22,51
(4) $SbC_4H_4NO_2$	RHF+grad.	*3-21G**	110	1	305,70	62,53
(5) *FOOF*	RHF+hessiana	*6-31G**	60	1	16,11	9,55
(6) SnC_5H_6	GVB-PP(3)	*3-21G**	96	6	690,11	371,04
(7) SiH_2	MCSCF	*6-31G***	29	51	11,27	19,21
(8) Si_2H_4	CI (2ª ordem)	*6-31G**	46	4.600	49,62	36,62
(9) C_3H_4	MCSCF+grad.	*DH(d)*	53	20	305,61	573,40
(10) O_2^+	Transição CI	*Duij(2d)*	60	504	17,21	26,36
(11) *OHBr*	MCSCF	*3-21G(d,p)*	49	110	257,69	297,33
(12) C_6H_8	UHF	*DH(d,p)*	130	1	306,24	114,60

[a]Orbitais Atômicos.
[b]Funções de correlação adaptadas à simetria.

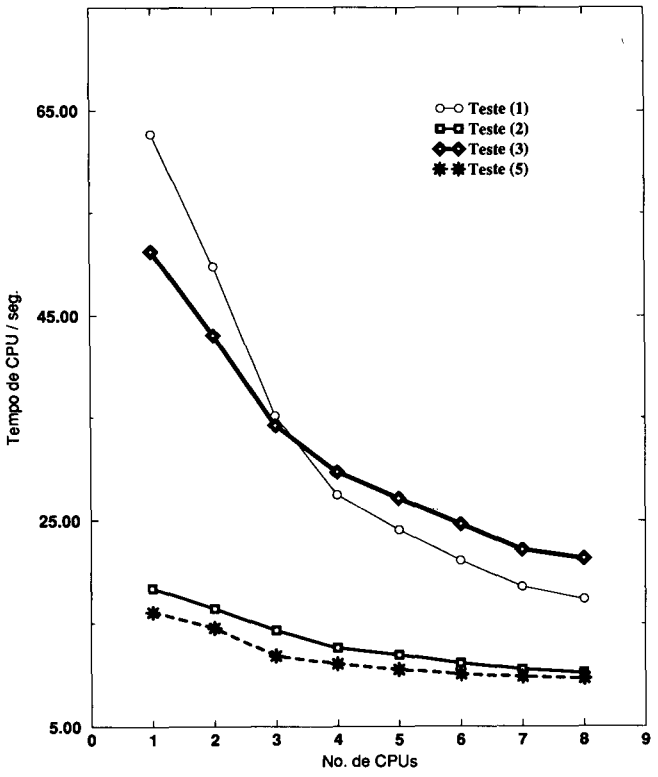

Figura 5. Tempo de CPU versus *número de processadores no SP1, para alguns cálculos de sistemas moleculares dados na tabela 5.*

(4) armazenando as integrais em disco, é o que fornece o menor tempo de execução, porém neste caso foram empregados os 8 processadores do computador SP1.

Do tempo de cálculo DIRETO com 1 processador para o tempo de cálculo paralelo DIRETO (com 8 processadores), houve uma redução de um fator 15, sendo que cada processador em seu disco local utilizou apenas 201,83 kb, contra os 1,2 Mb do primeiro caso. Esta é uma situação que pode ser utilizada quando da pouca disponibilidade de disco. Porém, a situação ótima observada, em termos de tempo de cálculo, é quando trabalha-se simultaneamente com os 8 processadores em paralelo e armazenamento das integrais em disco local (33,84 s), mas em compensação há um aumento em aproximadamente 30 vezes no uso de disco (de 201,83 kb para 6,02 Mb).

Através desta análise vale a pena salientar que deve-se trabalhar criteriosamente em função dos recursos computacionais disponíveis (número de processadores, quantidades de memória e disco, otimização dos códigos, etc), a fim de poder obter o melhor desempenho dos mesmos, considerando-se também a urgência na aquisição dos resultados em função da disponibilidade do tempo do pesquisador.

Tabela 6. Resultados de tempos de cálculos utilizando basicamente memória do processador ou disco para armazenar as integrais, obtidos através do programa GAMESS.

Teste	CPU (em seg)		Disco (em kb)	
	DIRETO	DISCO	DIRETO	DISCO
Teste (1)	62,72	17,02	61,93	5.655,65
Teste (4)	985,13	305,70	1.280,48	99.186,68
PAR[a]	62,53	33,84	201,83	6.024,55
Teste (6)	3.819,69	711,82	690,11	56.630,57

[a]Cálculo paralelo com 8 processadores. Seqüencial (1 processador) nos outros casos. Cada processador é uma RISC/6000 370 com 256 Mb de memória RAM.

CONCLUSÕES

O uso de técnicas de paralelização em uma rede de computadores permite reduzir substancialmente o tempo de execução de tarefas. Usando-se como modelo cálculos de integrais de repulsão eletrônica do método de orbitais moleculares, observou-se vantagens substanciais e ganho de tempo computacional, embora a eficiência desejada não seja tão explícita, de modo que nem sempre o aumento aleatório do número de CPUs, refletirá nestas vantagens.

Como em uma arquitetura de multicomputadores o número de processadores é geralmente pequeno em relação ao tamanho do problema que está sendo tratado, observou-se que os processadores trocam muitas mensagens, enquanto cooperam na solução do problema. Observou-se também que, em geral o tempo necessário para efetuar o tratamento de dados dentro de um processador é desprezível quando comparado ao tempo necessário para uma mensagem ir de um processador a outro. Porém, este tempo de transmissão via rede é insignificante, quando comparado ao tempo de solução do problema proposto, assim não há diminuição do desempenho da paralelização em termos das perdas de tempo de transmissão de mensagens.

Neste trabalho verificou-se a eficiência na transmissão de dados utilizando-se técnicas de paralelização para grandes conjuntos. Em conjuntos menores que 1Mb os tempos adicionais de comunicação entre máquinas não parece ser crítico em relação ao tempo de transmissão propriamente dito. A fragmentação destes pacotes de dados possibilita uma maior velocidade de transmissão.

Nesta análise, onde foram mostradas algumas potencialidades dos recursos computacionais paralelos e dos programas disponíveis para paralelização, obeservou-se que existe a necessidade de considerar-se alguns fatores para a obtenção do melhor desempenho computacional em função da necessidade do usuário, que pode ser feita observando o tempo de execução, disponibilidade de memória e disco, número de processadores, e assim obter a melhor relação custo/benefício.

Assim, além da apresentação de resultados e da eficiência da programação paralela, desejou-se com este trabalho fazer um breve apresentação de diferentes aspectos de paralelismo a fim de proporcionar referencial para o desenvolvimento de programação em ambientes que exigem intensa computação, uma vez que, principalmente multiprocessadores de memória distribuída e redes de estações de trabalho são seguramente plataformas promissoras para computação de alto desempenho.

AGRADECIMENTOS

O autor agradece ao Prof. Pedro A. M. Vazquez pelas sugestões e discussões relevantes para este trabalho e ao Prof. Rogério Custodio pela revisão e comentários feitos. Ao Instituto de Química e ao Centro Nacional de Processamento de Alto Desempenho (CENAPAD-SP) agradeço as facilidades computacionais. E, meu agradecimento também ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro.

REFERÊNCIAS E NOTAS

1. Hehre, W. J.; Radom, L.; Schleyer, P. R.; Pople, J. A.; *Ab initio* Molecular Orbital Theory, John Wiley & Sons, New York (1986).
2. Szabo, A.; Ostlund, N. S.; Modern Quantum Chemistry, Macmillan Pub. Co. Inc., New York (1982).
3. Harrison, R.; Bair, R.; *Theor. Chim. Acta*, (1993), **84**, 255.
4. Carbo, R.; Molino, L.; Calabuig, B.; *J. Comp. Chem.*, (1992), **13**, 155.
5. Kindermann, S.; Michel, E.; Otto, P.; *J. Comp. Chem.*, (1992), **13**, 414.

6. Rendell, A. P.; Lee, T. J.; Linddh, R.; *Chem. Phys. Letters*, (1992), **194**, 84.
7. Akl, S. G.; Parallel Sorting Algorithms, Academic Press Inc, Flórida, USA, (1985).
8. Baillie, F.; *Parallel Computing*, (1988), **8**, 101.
9. Bertsekas, D. P.; Tsitsiklis, J. N.; Parallel and Distribuited Computation, Prentice Hall, New Jersey, USA, (1989).
10. Whiddett, D.; Concurrent Programming for Software Engineers, Ellis Horwood Limited, London, (1989).
11. O neologismo "escalabilidade" (do inglês *scalability*) refere-se ao ato de expandir o número de processadores em uma rede, ou mais comumente em multicomputadores, produzindo um aumento proporcional na velocidade de processamento.
12. Pountain, D.; Brian, J.; *BYTE*, (1992), **112**, 113.
13. Feng, T.; *IEEE Computer* (1981), 18.
14. Informações ou aquisição destes *softwares* podem ser obtidas pelo correio eletrônico ou através de ftp anônimo, como segue: *titan.cs.rice.edu:* public/HPFF/draft/hpf-v10-final.tar.Z, informações sobre Pfortran com *scott@karazm.math.uk.edu* e *cs.ucla.edu:* pub/maisie.2.1.1.4.tar.Z, respectivamente.
15. O repositório do Instituto de Química/UNICAMP (143.106.13.1) no diretório pub/parallel contém alguns programas e ferramentas para paralelização. Mais especificamente podem ser obtidos onde indicado. HeNCE - *netlib@ornl.gov* send index from hence, ParaGraph - *netlibd@surfer.epm.ornl.gov* send index from paragraph, Xab - contato com *adam@cs.cmu.edu* e ADAPTOR - *ftp.gmd.de:* gmd/adaptor. Para os programas ver informações nas referências específicas.
16. Carriero, N.; Gelernter, D.; *ACM Computing Surveys*, (1989), **21**, 312. Obtido via ftp anônino em *nic.funet.fi:* pub/unix/parallel.
17. Pacote de *software* que permite a utilização de uma rede heterogênea de computadores paralelos e serial com recursos computacionais simples (versão 2.4.1). Desenvolvido por: Beguelin, A.; Dongarra, J.; Geist, A.; Manchek, R.; Sunderam, V. Produzido pelo Laboratório Nacional de OAK RIDGE, Oak Ridge, Tennessee, USA. Obtido por *NETLIB* em: *netlib@ornl.gov*, tendo no corpo da mensagem: send index from pvm3.
18. TCGMSG é um pacote utilizado na construção de programas paralelizados, usando um modelo *message passing*. Desenvolvido por: Harrison, R. J. Produzido pelo Grupo de Química Teórica do Laboratório Nacional de ARGONE, Illinois, USA. Disponível por ftp anônimo em *ftp.tcg.anl.gov:*pub/tcgmsg/tcgmsg.4.02.tar.Z.
19. Fox, G. C.; Johnson, M. A.; Lyzenga, G. A.; Otto, S. W.; Slamon, J. K.; Wlaker, D. W.; Solving Problems on Concurrent Processor, Vol. 1. Prentice-Hall International, (1988).
20. Hemped, R.; The ANL/GMD Macros (PARMACS) in FORTRAN for Portable Parallel Programming using Message Passing Programming Model, (1991).
21. Geist, G. A.; PICL a portable instrument communication library. Reference manual, Oak Ridge National Laboratory, (1990). Obtido em: *netlib@ornl.gov* send index from picl.
22. Buttlerm R.; Lusk, E.'; User's Guide to p4 Programming System, 9700 South Cass Avenue Argone, IL 60439-4801, (1992). Disponível por ftp anônimo em *info.mcs.anl.gov:* pub/p4/p4-1.4.tar.Z.
23. Allen, M. P.; *Theor. Chim. Acta*, (1993), **84**, 399.
24. Smith, W.; *Theor. Chim. Acta*, (1993), **84**, 385.
25. Kunz, A. B.; *Theor. Chim. Acta*, (1993), **84**, 353.
26. Guest, M. F.; Sherwood, P.; van Lenthe, J. H.; *Theor. Chim. Acta*, (1993), **84**, 423.
27. Hilbers, P. A. J.; Esselink, K.; *Proceedings of the European Workshops on Parallel Computing*, Barcelona, Espanha, (1992), 288.
28. Hedman, F.; Laaksonen, A.; *Int. J. Quant. Chem.*, (1993), **46**, 27.
29. Young, W. S.; Brooks III, C. L.; *J. Comp. Chem.*, (1994), **15**, 44.
30. Ripoll, D. R.; Thomas, S. J.; *J. Supercomp.*, (1992), **6**, 163.
31. Janak, J. F.; Pattnak, P. C.; *J. Comp. Chem.*, (1992), **13**, 1098.
32. Brower, R. C.; DeLisi, C.; *Crit. Rev. Biomed. Eng.*, (1992), **20**, 373.
33. Clementi, E.; Corongiu, G.; Detrich, J.; Chin, S.; Domingo, L.; *Int. J. Quantum Chem. Symp.* (1984), **18**, 601.
34. Clementi, E.; Corongiu, G.; Detrich, J.; *Comp. Phys. Comm.*, (1985), **37**, 287.
35. DISCO - Feyereisen, M; Kendall, R. A.; *Theor. Chim. Acta*, (1993), **84**, 289.
36. GAMESS - Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T. Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N. Nguyen, K. A.; Su, S. J.; Windus, T. L.; Dupuis, M.; Montgomery, J. A.; *J. Comput. Chem.*, (1993), **14**, 1347.Para obter uma cópia do programam escreva para Mike Schmidt no e-mail *mike@si.fi.ameslab.gov*.
37. HONDO - "The General Atomic and Molecular Electronic Structure System: HONDO 7.0", Dupuis, M.; Watts, J. D.; Villar, H. O.; Hurst, G. J. B.; *Comput. Phys. Comm.*, (1989), **52**, 415.
38. TURBOMOLE - Ahlrichs, R.; Bar, M.; Haser, M.; Horn, H.; Komel, C.; Electronic structure calculations on workstations computers. The program system on TURBOMOLE.
39. AMBER - Singh, U. C.; Weiner, P. R.; Caldwell, J. W.; Kollman, P. A; (1986) (UCSF versão 3.0). Dept. Pharmaceutical Chem., Univ. of California, San Francisco (USA).
40. Morgon, N. H.; Custodio, R.; *Quím. Nova*; (1995), **18**, 44.
41. Shirsat, R. N.; Limaye, A. C.; Gadre, S. R.; *J. Comp. Chem.*, (1993), **14**, 445.
42. Rendell, A. P.; Lee, T. J.; Komornicki, A.; Wilson, S.; *Theor. Chim. Acta*, (1993), **84**, 271.
43. Hollauer, E.; *Quím. Nova*; (1994), **17**, 301.
44. Colvin, M. E.; Jansen, C. L.; Whiteside, R. A.; Tong, C. H.; *Theor. Chim. Acta*, (1993), **84**, 301.
45. Lüthi, H. P.; Almöf, J.; *Theor. Chim. Acta*, (1993), **84**, 443.
46. Clarke, L. J.; *Theor. Chim. Acta*, (1993), **84**, 325.
47. Vogel, S.; Htter, J.; Fischer, T. H. Lütti, H. P.; *Int. J. Quant. Chem.*, (1993), **45**, 665.
48. Ludëk, M.; Koca, J.; *J. Comp. Chem.*, (1994), **15**, 937.
49. Versões posteriores (a partir da 3.0) possuem conceituação diferente a dada neste trabalho.
50. Cook, D. B.; "*Ab Initio* Valence Calculations in Chemistry", Flechter & Sons LTD., Norwich, (1974).

Publicação financiada pela FAPESP